SCHEMA METHODS

Example: Using Schema Methods for Simple Authentication

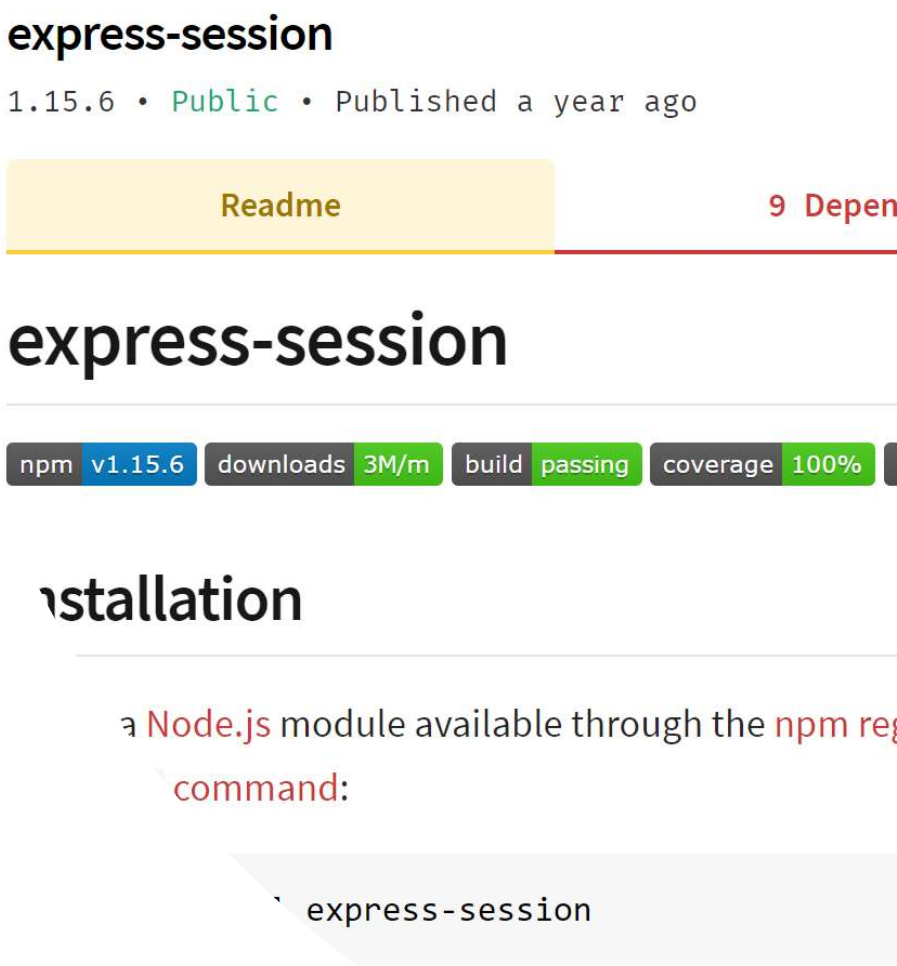
- Restrict access to Posts API (require authentication):
  - Create users schema with methods for
    - Finding users
    - Checking password
  - Use **express-session** middleware to create and manage user session (using cookies) FXW1
  - Create an authentication route to set up “session”
  - Create your own authentication middleware and place it on /api/posts route

Slide 34

FXW1 Frank X Walsh, 21/03/2019

Aside: Sessions

- Requests to Express apps are stand-alone by default
  - no request can be linked to another.
  - By default, no way to know if this request comes from client that already performed a request previously.
- Sessions are a mechanism that makes it possible to “know” who sent the request and to associate requests.
- Using Sessions, every user of you API is assigned a unique session:
  - Allows you to store state.
- The express-session module is middleware that provides sessions for Express apps.



User Schema with Static & Instance Methods

```
const UserSchema = new Schema({
  username: { type: String, unique: true, required: true},
  password: {type: String, required: true },
});

UserSchema.statics.findByUserName = function(username) {
  return this.findOne({ username: username});
};

UserSchema.methods.comparePassword = function (candidatePassword) {
  const isMatch = this.password === candidatePassword;
  if (!isMatch) {
    throw new Error('Password mismatch');
  }
  return this;
};

export default mongoose.model('User', UserSchema);
```

Static Method: belongs to schema. Independent of any document instance

Instance Method: belongs to a specific document instance.

express-session middleware

- Session middleware that stores session data on server-side
  - Puts a unique ID on client

```
npm install --save express-session
```

- Add to Express App middleware stack:

```
//session middleware
app.use(session({
  secret: 'ilikecake',
  resave: true,
  saveUninitialized: true
}));
```

Create User Route to authenticate

- Use **/api/user** to authenticate, passing username and password in HTTP body

/api/users/index.js

```
// authenticate a user, using async handler
router.post('/', asyncHandler(async (req, res) => {
    if (!req.body.username || !req.body.password) {
        res.status(401).send('authentication failed');
    } else {
        const user = await User.findByUserName(req.body.username);
        if (user.comparePassword(req.body.password)) {
            req.session.user = req.body.username;
            req.session.authenticated = true;
            res.status(200).end("authentication success!");
        } else {
            res.status(401).end('authentication failed');
        }

    }
}));
```

Using static method to find User document

Using instance method to check password

/index.js

```
app.use('/api/users', usersRouter);
```

Authentication Middleware

authenticate.js

```
import User from './api/users/userModel';
// Authentication and Authorization Middleware
export default async (req, res, next) => {
  if (req.session) {
    let user = await User.findByUserName(req.session.user);
    if (!user)
      return res.status(401).end('unauthorised');
    next();
  } else {
    return res.status(401).end('unauthorised');
  }
};
```

Checks for user ID in session object.
If exists, called next middleware function, otherwise end req/res cycle with 401

index.js

```
import authenticate from './authenticate';
```

```
app.use('/api/posts', authenticate, postsRouter);
```

Authentication middleware applied on /api/posts route.

Object Referencing

```
const PostSchema = new Schema({
  title: {type: String, required: true},
  link: {type: String, optional: true},
  username: {type: String, required: true},
  comments: [CommentSchema],
  upvotes: {type: Number, min: 0, max: 100, default: 0},
});
```

Using Object ID to reference user document

```
const PostSchema = new Schema({
  title: {type: String, required: true},
  link: {type: String, optional: true},
  user: {type: Schema.Types.ObjectId,
          ref:'User',
          required: true},
  comments: [CommentSchema],
  upvotes: {type: Number, min: 0, max: 100, default: 0},
});
```

Query Population using Refs

- Allows you to automatically replace the specified paths in the document with document(s) from other collection(s).

```
async function refTest() {
    const user1 = new User({
        username: "user99",
        password: "pass1"
    });
    await user1.save();


    const post1 = new Post({
        title: "A Post",
        user: user1._id
    });

    await post1.save()
    Post.find({})
        .populate('user')
        .exec(function (error, posts) {
            console.log(JSON.stringify(posts, null, "\t"))
        });
}

refTest();
```

output

```
{
        "upvotes": 0,
        "_id": "5c93899a1f4eaa3cf4e4fbc8",
        "title": "A Post",
        "user": {
                "_id": "5c9389981f4eaa3cf4e4fbc7",
                "username": "user99",
                "password": "pass1",
                "__v": 0
        },
        "comments": [],
        "__v": 0
}
```

MONGODB AS A SERVICE

MongoDB as a Service

- Best practice for initial development is to host MongDB process on your development machine
- In production environments, Mongo will be hosted:
  - on it's own instance or
  - provisioned as a service

MongoDB as a Service

MongoDB Atlas

Move faster with an automated cloud MongoDB service built for agile teams who'd rather spend their time building apps than managing databases. Available on AWS, Azure, and GCP.

Start free

Already have an account? **Log in here →**

Cloud Provider & Region

Choose your preferred cloud provider and the region nearest to clients

AWS, N. Virginia (us-east-1)

Select a cloud provider to see its region availability.

aws | Google Cloud Platform | Azure

Configure a **free tier cluster** by first selecting a region labeled with FREE TIER AVAILABLE then choose the M0 option in the Cluster Tier below.

★ recommended region

NORTH AMERICA	EUROPE	ASIA
N. Virginia (us-east-1) ★ FREE TIER AVAILABLE	Ireland (eu-west-1) ★	Tokyo (ap-northeast-1)
Ohio (us-west-1) ★	London (eu-west-2)	Seoul (ap-northeast-2)
N. California (us-west-1)	Frankfurt (eu-central-1) ★ FREE TIER AVAILABLE	Singapore (ap-southeast-1)
Oregon (us-west-2) ★		Mumbai (ap-south-1)
Montreal (ca-central-1)	SOUTH AMERICA: São Paulo (sa-east-1)	

Pricing | Getting Started | Migrate to MongoDB Atlas | Frequently Asked Questions

MongoDB as a Service

- Some providers allow free access tier
- Provide user credentials wrapped in a URL
- All you need to do is update your config with the relevant URL
- Again, be careful to ignore credentials when pushing to github/public repo